Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht ausgewiesene Kompetenzbereiche für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzbereiche in engem Bezug zueinanderstehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzbereiche				
	K1	K2	K3	K4	K5
1.1			X		
1.2.1	X				
1.2.2				X	
1.3.1		X	X		
1.3.2				X	
1.4			X		
1.5.1		X	X		
1.5.2				X	
2.1	X				X
2.2	X				
2.3			X		
2.4.1				X	
2.4.2				X	
2.4.3				X	
2.4.4				X	
2.5		X	X		

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Datenbanksysteme
verbindliche Themenfelder: Objektorientierte Modellierung (Q1.1), Implementierung von Klassen und Assoziationen (Q1.2), Konzeptionelle und logische Modellierung einer Datenbank (Q2.1), Datenabfrage und Datenmanipulation mit SQL (Q2.2)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Nicht für den Prüfling bestimmt

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
1.1	entwickeln, zeichnen Kunde; Buchungssystem Ferienhäuser; anmelden; freie Ferienhäuser suchen, extension points: Buchung; <<extend>>; Condition: {Buchung gewünscht} extension point: Buchung; Ferienhaus buchen; <<include>>; Buchung stornieren; Benutzer registrieren; Belegung anzeigen; Ferienhaus einstellen; Preise ändern; Administrator; Vermieter			
	entwickeln		3	2
	zeichnen	3		

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.2.1	beschreiben Die Klasse `Ferienhaus` besitzt die privaten Attribute `nr`, `maxPersonen` als ganze Zahlen, `tagesPreise` als Array von ganzen Zahlen und `adresse` als Objekt der Klasse `Adresse`, zudem ein Klassenattribut namens `autowert`, das dazu dient, für jedes Ferienhaus eine eindeutige `nr` zu vergeben. Dies übernimmt der Konstruktor, der auch die Instanzvariablen mithilfe der Parameter und die Beziehungsattribute initialisiert. Die Klasse `Ferienhaus` verfügt über die öffentlichen Methoden `ermittlePreis()`, `hinzufuegenBuchung()` und `istFrei()`. Die Methode `ermittlePreis()` hat beispielsweise als Parameter Referenzen auf zwei Datumsobjekte. Sie liefert eine Zahl vom Typ `double` zurück. Ein Ferienhaus kann mehrmals gebucht (`buchungen`) werden, deswegen hat die Beziehung zwischen Ferienhaus und Buchung die Multiplizität `0..*`. Jede Buchung ist genau einem Ferienhaus zugeordnet (Multiplizität `1`) und kann von der Buchung über die Rolle `objekt` referenziert werden. Darüber hinaus ist jedes Ferienhaus einer Region (`region`) zugeordnet. Beide Assoziationen sind beidseitig navigierbar. Die Beziehung zur Klasse `Verwaltung` ist nur einseitig navigierbar, d.h. Ferienhausobjekte kennen die Verwaltung nicht.	2	2	
1.2.2	überführen, implementieren (Code siehe unten)			

```java
public class Ferienhaus {
  private static int autowert = 0;
  private int nr;
  private int maxPersonen;
  private int[] tagesPreise;
  private Adresse adresse;
  private List<Buchung> buchungen;
  private Region region;

  public Ferienhaus(Region region, Adresse adr, int maxP,
                    int[] tagesPreise) {
    this.nr = ++autowert;
    this.region = region;
    this.adresse = adr;
    this.maxPersonen = maxP;
    this.tagesPreise = tagesPreise;
    buchungen = new List<>();
    region.hinzufuegenObjekt(this);
  }
  public void hinzufuegenBuchung(Buchung buchung) {
    boolean positionGefunden = false;
    int index = 0;
    for (; index < buchungen.size() && !positionGefunden;
         index++) {
      Buchung b = buchungen.get(index);
      if (b.getVon().isAfter(buchung.getVon()))
         positionGefunden = true;
    }
    if (positionGefunden)
      index--;
    buchungen.add(index, buchung);
  }
```

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td></td><td>

```java
  public boolean istFrei(Date von, Date bis) {
    boolean ergebnis = true;
    for (Buchung b : buchungen) {
      if (b.getVon().isBetween(von, bis) ||
        b.getBis().isBetween(von, bis) ||
        (b.getVon().isBefore(von) &&
         b.getBis().isAfter(bis))
      {
        ergebnis = false;
        break;
      }
    }
    return ergebnis;
  }
}
```

überführen
implementieren</td><td>
2</td><td>

3</td><td>

3</td></tr>
<tr><td>1.3.1</td><td>entwickeln, überführen

double ermittlePreis(vonDatum, bisDatum)

```
vonMonat := ermittle Monat aus vonDatum
bisMonat := ermittle Monat aus bisDatum
vonMonat = bisMonat ?
J:
  tage := Tag des bisDatums - Tag des vonDatums
  preis := tage * tagesPreise[vonMonat-1]
N:
  tage := Tage im Monat des vonDatums - Tag des vonDatums
  preis := tage * tagesPreise[vonMonat-1]
  tempDatum := vonDatum + 1 Monat
  tempMonat := Monat von tempDatum
  solange (tempDatum < bisDatum und
           tempMonat != bisMonat)
    tage := Anzahl Tage im Monat tempDatum
    preis := preis + tage * tagesPreise[tempMonat-1]
    tempDatum := tempDatum + 1 Monat
    tempMonat := Monat von tempDatum
  tage := Tag von bisDatum
  preis := preis + tage * tagesPreise[bisMonat-1]
Rückgabe: preis
```

überführen</td><td>
3</td><td>3</td><td>4</td></tr>
</table>

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.3.2</td><td>überführen
<pre>
  public double ermittlePreis(Date von, Date bis) {
    double preis = 0.0;
    int tage = 0;
    int vonMonat = von.getMonat();
    int bisMonat = bis.getMonat();
    Date tempDatum = null;
    int tempMonat = 0;

    if (vonMonat == bisMonat) {
      tage = bis.getTag() - von.getTag();
      preis = tage * tagesPreise[vonMonat-1];
    } else {
      //Start Monat
      tage = von.getNumberOfDaysInMonth() - von.getTag();
      preis = tage * tagesPreise[vonMonat-1];
      // Zwischenmonate
      tempDatum=von.addMonths(1);
      tempMonat=tempDatum.getMonat();
      while (tempDatum.isBefore(bis) &&
             tempMonat() != bisMonat) {
        tage = tempDatum.getNumberOfDaysInMonth();
        preis += tage * tagesPreise[tempDatum.getMonat()-1];
        tempDatum=tempDatum.addMonths(1);
        tempMonat= tempDatum.getMonat();
      }
      // End Monat
      preis += bis.getTag() * tagesPreise[bisMonat-1];
    }
    return preis;
  }
</pre></td><td>3</td><td>3</td><td></td></tr>
</table>

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
1.4	entwickeln, zeichnen			
	entwickeln		4	2
	zeichnen	4		

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.5.1	modellieren		5	3
1.5.2	implementieren		3	3

1.5.1 modellieren

<<abstract>>
Person
- autowert : int
- nr : int
- name : String
- vorname : String
- adresse : Adresse
+ Person(name : String, vorname : String, adresse : Adresse)

Kunde
- hatHaustier : boolean
- istRaucher : boolean
+ Kunde(nn : String, vn : String, adr : Adresse)
+ stornieren(buchungsNr : int) : boolean

Vermieter
- iban : String
+ Vermieter(nn : String, vn : String, adr : Adresse, iban : String)

Buchung

– kunden *
1 – kunde
0..* – buchungen
– vermieter 1
– vermieter *
0..* – buchungen
– objekt 1
– objekte 1..*

Ferienhaus
- haustierErlaubt : boolean
- rauchenErlaubt : boolean
- ... : vorhandene Eigenschaften
+ Ferienhaus(reg : Region, adr : int, maxP : int, tagesPreise : int, v : Vermieter)
+ zeigeBelegung(jahr : int) : boolean[][]
+ aendernPreise(tagesPreise : int[])
. . . vorhandene Methoden

– objekte *

Verwaltung
+ registrierenKunde(name : String, vorname : int, adr : Adresse) : Kunde
+ registrierenVermieter(name : String, vorname : String, adr : Adresse) : Vermieter
+ anmeldenKunde(name : String, vorname : String) : Kunde
+ anmeldenVermieter(name : String, vorname : String) : Vermieter
. . . vorhandene Methoden

1.5.2 implementieren

```java
public abstract class Person {
  private static int autowert = 0;
  private int nr;
  private String name;
  private String vorname;
  private Adresse adresse;

  public Person(String name, String vorname, Adresse adr) {
    this.nr = ++autowert;
    this.name = name;
    this.vorname = vorname;
    this.adresse = adr;
  }
}
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III

	<pre>public class Vermieter extends Person {
	  private String iban;
	  private List<Ferienhaus> objekte;
	
	  public Vermieter(String name, String vorname, Adresse adr,
	                   String iban) {
	    super(name, vorname, adresse);
	    this.iban = iban;
	  }
	}</pre>			

	Summe 60	**17**	**26**	**17**

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.1	beschreiben Eine Relation befindet sich in der 1. NF, wenn alle Attribute nur atomare Werte aufweisen und ein Primärschlüssel existiert, der die Eindeutigkeit der Datensätze gewährleistet. Eine Relation befindet sich in der 2. NF, wenn sie in der 1. NF ist und jedes Nicht-Schlüsselattribut vom Primärschlüssel voll funktional abhängig ist. Es ist zu beachten, dass die Attribute nicht nur von einem Teilschlüsselfeld, sondern vom gesamten Schlüssel abhängig sind. Eine Relation befindet sich in der 3. NF, wenn sie in der 2. NF ist und jedes Nicht-Schlüsselattribut nicht transitiv vom Primärschlüssel abhängig ist. Transitive Abhängigkeit bedeutet, dass ein Nicht-Schlüsselattribut nur auf Umweg über ein anderes Nicht-Schlüsselattribut vom Primärschlüssel abhängig ist.	3		
	prüfen Die Tabellen Person und Ferienhaus liegen nicht in der 1. Normalform, da zum Beispiel die Werte in den Datenfeldern Adresse nicht atomar sind. Die Tabelle Belegung befindet sich in der zweiten Normalform, da alle Werte atomar sind und alle Nicht-Schlüsselattribute vollständig vom Primärschlüssel abhängig sind. Die Tabelle Belegung verstößt gegen die 3. Normalform, da die Typbezeichnung funktional abhängig ist von der Typnummer und somit nur transitiv vom Primärschlüssel ID abhängt.		3	
2.2	beschreiben Änderungsanomalie: Redundanzen können zu Inkonsistenzen führen, wenn die Änderung eines redundant gespeicherten Wertes nicht vollständig erfolgt. In der Tabelle Ferienhaus besteht bereits eine Inkonsistenz, da das Land mit dem Code FR die beiden Typbezeichnungen Frankreich und France aufweist. Einfügeanomalie: Beim Einfügen von Daten in eine Datenbank spricht man von einer Einfügeanomalie, wenn ein neues Tupel in die Relation nicht oder nur schwierig eingetragen werden kann, weil nicht zu allen Attributen des Primärschlüssels Werte vorliegen oder viele Spalten null-Werte aufweisen würden. In die Tabelle Belegung kann kein neuer Belegungstyp aufgenommen werden, so lange keine Belegung dieses Typs existiert.	3		

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE		
		I	II	III
	Löschanomalie: Es gehen durch das Löschen eines Datensatzes mehr Informationen verloren als gewünscht. Wenn der 4. Datensatz der Tabelle Belegung gelöscht wird, geht auch der Belegungstyp Gesperrt verloren.			

2.3	überführen
	`Person(personNr, nname, vname, strasse, plz, ort, email)`
	`Land(code, land)`
	`Region(regionNr, name, laenderCode#)`
	`Ferienhaus(objektNr, bezeichnung, strasse, plz, ort, regionNr#, maxPersonen, besitzer#)`
	`Belegungstyp(typNr, bezeichnung)`
	`Belegung(ID, objektNr#, personNr#, von, bis, typNr#)`
		4	2	

2.4.1	formulieren
	`INSERT INTO Person(nname, vname, strasse, plz, ort)`
	`VALUES ('Dahl', 'Klaus', 'Am Lauberg 17', '36037', 'Fulda');`
	
	`INSERT INTO Ferienhaus(bezeichnung, strasse, plz, ort, regionNr, maxPersonen, besitzer)`
	`VALUES ('Villa Waldesruh', 'Forstweg 27', '36145', 'Hofbieber', 2, 6, (`
	`SELECT personNr FROM Person`
	`WHERE nname = 'Dahl' AND vname = 'Klaus'));`
			4	

2.4.2	entwickeln
	`SELECT b.personNr, b.von, b.bis, bt.bezeichnung`
	`FROM Belegung b JOIN BelegungsTyp bt USING(typNr)`
	`WHERE objektNr = 100`
	`AND bis > NOW()`
	`ORDER BY von;`
			2	1

2.4.3	entwickeln
	`SELECT l.land, r.name, COUNT(*) AS 'Anzahl Ferienhäuser'`
	`FROM Ferienhaus f JOIN Region r ON f.regionNr = r.regionNr`
	`JOIN Land l ON r.laenderCode = l.code`
	`GROUP BY l.land, r.name;`
			1	2

2.4.4	implementieren
	`SELECT *`
	`FROM Ferienhaus`
	`WHERE regionNr = 1`
	`AND maxPersonen BETWEEN 4 AND 6`
	`AND objektNr NOT IN`
	`(SELECT objektNr FROM Belegung`
	`WHERE`
	`(von BETWEEN '2024-05-11' AND '2024-05-18' OR`
	`bis BETWEEN '2024-05-11' AND '2024-05-18')`
	`OR (von < '2024-05-11' AND bis > '2024-05-18')`
	`);`
			2	2

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.5	modellieren, zeichnen Hinweis: Eine Lösung ohne schwachen Entitätstyp ist als gleichwertig zu betrachten. modellieren zeichnen	 2	 3	 6
	Summe 40	**12**	**17**	**11**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“, „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/ schwerpunktbezogene Fächer) (Abiturerlass BG)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Praktische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	17	26	17	**60**
2	12	17	11	**40**
Summe	**29**	**43**	**28**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.